Computer Science • 19(2) 2018 https://doi.org/10.7494/csci.2018.19.2.2844

Michał Wrzeszcz
Renata G. Słota
Jacek Kitowski

TOWARDS TRANSPARENT DATA ACCESS WITH CONTEXT AWARENESS

Abstract *Open-data research is an important factor accelerating the production and analysis of scientific results as well as worldwide collaboration; still, very little data is being shared at scale. The aim of this article is to analyze existing data-access solutions along with their usage limitations. After analyzing the existing solutions and data-access stakeholder needs, the authors propose their own vision of a data-access model.*

Keywords distributed, transparent data access, distributed data storage, context awareness

Citation Computer Science 19(2) 2018: 201–221

1. Introduction

Data access and management can be analyzed on several levels; from personal data to globally distributed shared data with different problems to be overcome by data access and management tools. The simplest case is access to local data; i.e., data stored on direct attached storage (DAS), where the focus is on providing the device drivers and solutions that use the hardware optimally.

The next level is to provide data access for a group of users working for a single organization; e.g., provision of network attached storage (NAS). The problems encountered in this case pertain to possible network failures, higher latencies [23], or the simultaneous work of many users influencing the quality of service [38, 53]. The balance of use of available storage systems to maintain QoS and cost effectiveness in an organization while request scheduling and resource allocation is also expected [9, 50].

The resources offered by a single organization can be insufficient for some users who need large amounts of storage and computing power to process the streams of data continuously produced by experiments or to make use of the large amount of information gathered in various existing datasets. To simplify the use of resources that belong to many organizations, the resource providers create federated organizations (FO), often defining a storage attached network (SAN) and detailing common rules of cooperation and resource sharing [35]. While grids [17] and virtual organizations (VOs) [30] introduce issues of decentralized management by organizations that use different policies and make autonomous decisions according to the local requirements, further work is required to improve the efficiency and convenience of data access as well as cost-effective data management.

The most complex case is the use of data provided by several nonfederated organizations (NFOs); i.e., organizations that do not have any agreement of cooperation. In this case, the challenges connected with trust and a lack of standards appear. As there is no bond between NFOs, the exchange of information about users and their data is difficult, as each NFO can use its own authentication mechanism.

While a cumulative increase of difficulty is observed at each level, modern science introduces a lot of problems that require the joint work of teams as well as large amounts of resources to solve the problem.

Concepts such as Open Science [37], Science 2.0 [16], and Data Science [12] require crossing the boundaries of a single organization. In such a case, there is a strong need for supporting data storage and access as well as the management of resources. This need is also connected with the fourth paradigm, which is believed to be one of the most important research paradigms nowadays [25]. Traditional data processing applications and database management tools may not be sufficient to support modern complex research, where the processing of Big Data [36] and cooperation that crosses organizational boundaries have become especially important.

2. Problem identification

Nowadays, more and more scenarios require high-performance data access to huge volumes of distributed data to process data-intensive scientific applications (e.g., astronomy data [31]) and/or share large experimental data and simulation results between groups of researchers (e.g., Human Brain Project [26] or Worldwide LHC Computing Grid [63]). However, despite the fact that the quality of the tools dedicated for data storage, access, and management is growing, most improvements address data access inside a single organization; consequently, working with data distributed at the resources that belong to several organizations is still difficult, resulting in the low sharing of data [5].

It has been noticed before that extracting knowledge or insights from data can lead to a better understanding of contemporary (scientific) problems [24]. The needs of building e-infrastructures dealing with open data, easy sharing, and access in organizationally distributed environments has resulted in the start of various projects and initiatives; e.g., [28] and [15].

The aim and scientific contribution of this article are defined as follows:

- an analysis of existing data-access solutions in order to identify reasons why so little data is being shared despite increased profits;
- a proposition of developing a data-access model based on the results of an analysis of the existing data-access solutions.

According to the CAP theorem [20], it is impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees: consistency, availability, and partition tolerance. Supporting Open Science, the processing of Big Data, and cooperation of users working on the resources of different organizations, it is important to provide availability and partition tolerance. This statement can be justified by a use case where several scientists process different parts of a read-only dataset; e.g., they analyze different scans of the human brain. Analysis of each part of a dataset takes a lot of time and requires a lot of computational resources, so it is conducted at several computing centers. Although the analysis of a particular data part does not require access to the full dataset, the results of the analysis should be available for anyone for a further comparison of the results obtained with different parts of the dataset. As the processing of each data part can take days or weeks, it is critical to allow us to process chosen data parts, even when access to other data parts is temporary impossible or the connection to other computing centers that take part in the processing is lost. For such a use case, it is important to finally provide access to the results produced using multiple data parts, not necessarily the provision of a consistent view on temporary data created during the processing. Thus, the authors focus on tools that allow the efficient accessing of data anytime rather than consistency. Tools like GlobalFS [40] will be not considered in this article, as they prefer to ensure strongly consistent file system operations despite node failures at a price of possibly reduced availability.

3. Different perspectives on data access

There are three main data-access stakeholders: users, providers (organizations that own/operate computing and/or storage resources and provide them to the user), and developers. The user expects a set of particular functionalities, while the provider tries to satisfy the user's requirements with its limited resources. The developer creates services that provide functionality for the user. These services represent IT platforms or tools to support use cases typical for a given scientific area. Thus, the users and providers are the most important stakeholders. The developer can be perceived as an advanced user that requires additional functionality to allow integration of the created services with the data-access system.

The main issues from the user's perspective are depicted below:

- easy anytime/anywhere data access,
- easy data sharing,
- efficient access to large volumes of data,
- archiving or making data public,
- advanced control over data storing,
- data dynamics,
- data security.

The provider's point of view is strictly related to the expectations of the users – they want to have data access realized efficiently, easily, and safely. They manage the data and resources to provide the users with the greatest benefits possible. However, since the storage systems used by each provider differ in terms of speed, capacity, and cost of purchase and maintenance, the providers require the ability to influence automatic data management in order to tune the management algorithms to their resources. The providers also expect the monitoring and management of resource usage for fair resource sharing to prevent the user quality of service from unjustified usage with high demand.

The cooperation of providers results in additional security and operational aspects. Although the providers often agree for some cooperation rules (e.g., forming a federation), they usually require full autonomy of their own resource management and access control.

On the basis of the issues described above, the following aspects of existing data-access solutions have been analyzed:

- efficiency, scalability, and limits on storage size or transfer speed,
- ease of access, including anytime/anyplace access to data from one place using convenient interface,
- ability to provide decentralized management,
- elasticity allowing easy integration with existing infrastructure.

Ease of use is difficult to define; however, when the data is distributed over several storage systems, the manual management of data locations, replicas, etc. is difficult for many users. Thus, the important term in an analysis of the solutions is data-access transparency. Transparent data access is defined by using a virtual path or identifiers with the automatic management of data locations and replicas. All problems with different data formats, storage systems, and data locations are hidden from the users in order to call data access transparent.

4. Analysis of existing data-access solutions

This section includes an analysis of various types of solutions:

- typical grid and cloud data-access tools,
- tools for anytime/anyplace data access,
- tools for distributed data processing,
- tools for unified view of multi-organizational data.

4.1. Typical grid and cloud data-access tools

Grid and cloud providers offer storage systems for different purposes. The grid usually supports solutions like (1) scratch for intermediate job results and data processing, (2) storage for final job results and long-term data storage accessible through a dedicated API, appropriate for data sharing between different sites. The providers can offer also (3) object storage that manages data as objects (e.g., Amazon S3). The access to an object is fast and scalable, but there is no hierarchical structure nor block access like in traditional file systems. Object storage is designed to deliver many online storage services, whereas the traditional storage systems are primarily designed for high-performance computing and transaction processing.

The selected examples of tools used in the grid and cloud environments are outlined below. Lustre [32] is a parallel distributed file system for computational clusters. The Lustre file system is often used as a high-performance scratch system in the grid. In such a case, there are usually different Lustre instances on the different sites, which means that the data stored on this file system can only be shared within the local cluster. Although the efficiency of the Lustre system is high, there is still a possibility of improving it using dedicated tools such as QstorMan [50–52].

In order to make the data available outside the site, it should be copied to a permanent storage outside the local cluster; e.g., the LFC (LCG File Catalog), which is the storage software for metadata management that provides common file system functionality for distributed storage resources [4]. It supports file replication for better data protection and availability. It is commonly used with the [63] command line utilities. Direct access from the application's source code is possible using GFAL API [22]. Since many users consider the usage of dedicated command line utilities or dedicated APIs as a drawback, they may use a FUSE-based [18] implementation of the file system called GFAL-FS [22], which provides access to data in the same manner as in the regular Unix-like file system. This type of access is slower (compared to the previous two methods) and only supports the read-only mode.

The OpenStack Object Store (known as Swift [39]) is an example of object storage often used in the cloud. It is able to provide common file names within the grid and cloud infrastructures, so it can be used in similar use-cases as in the LFC. However, the Swift file-sharing mechanism (which makes the use of an API access key-sharing or a session token) seems to be more difficult for most of users to use than the LFC file-sharing mechanism based on Unix permissions.

The standard grid and cloud environments do not offer data-access transparency when deployed in a multi-organization environment due to the heterogeneity of the storage systems (resulting from the different sets of tools used on different sites). The lack of easy transparent data access results in management problems from the provider's point of view. Less technically advanced users often work only with scratch storage and manually manage data transfers using SSH-based protocols for both file sharing and staging before job execution. This results in the non-optimal usage of storage and computing resources. Thus, both users and administrators require better data-access methods.

4.2. Tools for anytime/anyplace data access

Currently the existing tools for anytime/anyplace data access focus on the ease of access. The most popular ones are Dropbox, OneDrive and Google Drive [33]. Client applications are provided for the most popular operating systems allowing mounting a virtual file system that transparently handle synchronization with the cloud storage. If any operation performed without connection to the Internet conflicts with changes at the server side made by other clients, the user can resolve the conflict on his own. Other significant features are file-sharing mechanisms, which allow the users to easily publish their data.

These tools impose rigorous limits on the storage size and transfer speed, which become an obstacle when the research is conducted in geographically distributed manner and the data requires synchronization on-line between sites. The user has to carefully plan data processing according to the place of data creation and transfers/synchronization operations.

Another similar sync-and-share tool is ownCloud [34]. It enables the users to maintain full control over data location and transfer, while hiding the underlying storage infrastructure, abstracting file storage available through directory structures or WebDAV. It also provides file synchronization between various operating systems, sharing of files using public URLs, and support for external cloud storage services. Although ownCloud is more flexible than the previously mentioned tools, its performance is also not sufficiently high for data-intensive applications.

4.3. Tools for distributed data processing

One of the most prominent tools for remote data access is Globus Connect [2]. It is built on the GridFTP protocol to provide fast data movement and data-sharing capabilities inside an organization. Globus Connect focuses on data transfer and does not abstract the access to existing data resources. Thus, it does not provide any data-access transparency.

Another possibility in distributed environments is the provision of storage resources through a high-performance parallel file system. Solutions of this type intend to provide access to storage resources optimized for performance. They are usually built on top of dedicated storage resources (e.g., RAIDs), and they expose a POSIX-compliant interface. Examples of such solutions include BeeGFS (formerly FhGFS) [3], GlusterFS [21], Coda [6], and PanFS [42]. There are significant differences between these systems in terms of data access. The most important features of these systems are presented below.

BeeGFS [3] is an excellent example of a high-performance parallel file system because it uses many typical mechanisms for this type of tool. It combines multiple storage servers to provide a shared network storage resource with striped file contents. Built on scalable multithreaded core components with native Infiniband support, BeeGFS has no architectural bottlenecks. It strips file contents across multiple storage servers and distributes metadata of a distributed file system across multiple metadata servers. This results in high availability and low metadata-access latency. Even with a multiplication of metadata servers, it is guaranteed that changes to a file or directory by one client are immediately visible to other clients. BeeGFS has no support for the integration of resources managed by several independent organizations. It can be used within a local storage area, but it is difficult to provide transparent data access for organizationally distributed environments.

An interesting alternative for metadata servers is presented by GlusterFS. It uses an elastic hashing algorithm that allows each node to access data without use of metadata or location servers. The storage system nodes have the intelligence to locate any piece of data without looking it up in an index or querying another server. This parallelizes data access and ensures good performance scaling. GlusterFS can scale up to petabytes of storage available to the user under a single mount point.

The GlusterFS authors indicate the use of the elastic hashing algorithm as the heart of Gluster's fundamental advantages, which results in good performance, availability, stability, and a reduction in the risk of data loss and data corruption or the data becoming unavailable. The use of hashing algorithms minimizes traffic flow; however, usage of the metadata servers results in better elasticity and easier reconfiguration. On the other hand, the hashing algorithms require more work when a group of data servers is reconfigured, so both solutions have pros and cons; one's choice of solution should depend on the use case. Similar to BeeGFS, GlusterFS is rather suitable for a local storage area with no strong support for a distributed environment.

Coda [6] is an example of a system with strict support for disconnected mode operations. It offers a high availability of files by replicating a file volume across many servers and caching the files at the client machine. The server and client communicate through Remote Procedure Calls (RPC). The server keeps sending back messages to working clients. When a server is notified of file updates, it informs the clients that are caching a copy of it to invalidate that copy. However, without client-side replicas of the data, the user is not able to work in the case of a network failure, and the aggressive caching lead to conflicts. Automatic conflict resolving may lead to a loss of data (the user may be not aware of the problem) while manual conflict resolving is inconvenient for the users. The main drawback of Coda is its lack of support for the organizational distribution of an environment. Even if multiple organizations would be able to divide control over the Coda servers among themselves, the used cache coherency algorithm can result in the high utilization of the network between sites or in those cases when the cache of a particular client is not invalidated when it should be.

Another interesting tool is PanFS [42] that creates a single high-performance pool of storage under a global namespace. While most of the storage systems loosely couple the parallel file system software with legacy block storage arrays, PanFS combines the functions of a parallel file system, volume manager, and RAID engine into one holistic platform. It also makes efficient use of SSD storage to improve its performance. It is a commercial tool dedicated to building high-performance storage solutions for a single organization. Thus, similar to the solutions mentioned above, it is suited for local storage area usage but not appropriate for data access in distributed environments.

It is also worth mentioning the solutions built on the top of object storage systems, like CephFS [8], DDN's WOS [60], and Scality Ring [48]. While CephFS provides a POSIX-compliant distributed file system based upon RADOS [61], WOS delivers only true object storage with no underlying file system. The architecture of WOS consists of three components: building blocks, WOS Core software, and a choice of simple interfaces. The backend of a WOS storage infrastructure are the WOS storage nodes, which are essentially servers filled with 60 SATA disks each. The WOS Core has self--healing capabilities and a single management console for the entire infrastructure. The distribution of data in WOS may be configured to use geographic replication for disaster protection. Scality Ring is a software-only storage solution built using a distributed shared-nothing architecture. A distinguishing feature of this tool is its built-in tiering that provides high flexibility in storage configuration. Unfortunately, none of the tools offer transparent data access when deployed over distributed resources managed by several organizations.

Hadoop Distributed File System (HDFS) [49] offers support for the map-reduce paradigm, which is another direction of effective distributed data processing. It is designed to stream large data sets at high bandwidth for the user's processes. The data and metadata are stored separately. The system scales up by adding more servers. HDFS benefits from replication data among the available storage resources, but the metadata server can be considered a single point of failure that decreases the level of fault-tolerance of the system. Another interesting tool that supports the map-reduce paradigm is Tachyon [56], which provides high performance for map-reduce applications by aggressively using memory. Neither HDFS nor Tachyon provide for functionality of the transparent data access in a federation. Although HDFS supports federations, the purpose of an HDFS federation is to improve scalability and isolation but not to hide data distribution from the user.

The above-mentioned tools differ in implementation, which influences their non-functional features. GlusterFS uses the elastic hashing algorithm, while CephFS uses a metadata server and Scality RING utilizes a routing-based algorithm within a P2P network. The presented systems have different approaches for working in offline mode and caching. Moreover, dedicated hardware is required for some of them. It increases the efficiency of the system for the cost of investment.

Despite the variety among the systems, these tools are not suited well for organizationally distributed environments due to their limited support for provider cooperation due to their centrally managed storage systems. They are able to provide transparent data access only inside a single organization. Moreover, most of the above-mentioned tools are also difficult to use because of the limited support for deployment, with the resources already storing some data.

4.4. Tools for unified view of multi-organizational data

Another type of solution for exposing storage resources are systems that aim at providing an abstraction layer on top of the storage resources across multiple organizations. They can provide a coherent view of the user data stored in different systems. These systems expose a single namespace for data storage and often facilitate data management by enabling administrators to define the data-management rules. An exemplary tool is iRODS [27,47] developed for grid environments. The data can be simply stored at designated folders on any number of data servers. To integrate various external data-management systems such as GridFTP-enabled systems, SRM-compatible systems, and Amazon's S3 service, a plug-in mechanism can be used.

Data integration in the iRODS system is based on a metadata catalogue – iCAT – that is involved in the processing of the majority of data-access requests. Metadata information about the actual data stored in the system contains information like filename, size, location, and user-defined metadata as well. The user can search for data that has been tagged while the administrator can query the metadata catalogue directly by using an SQL-like language to provide aggregated information about the system. Although iCAT provides great functionality for both users and administrators, it is also considered to be a weakness of the system. It is implemented as a relational database, potentially a bottleneck of the whole system and a single point of failure.

Due to iRODS's ability to adjust data management to administrators/users needs, it is often referred to as an adaptive middleware. To allow for dynamic adaptation, the iRODS system uses rules. A rule is a chain of activities provided by low-level modules (built in or supported externally) to provide the required functionality. The users' actions are monitored by the rule engine to activate the rules. Typical user interfaces are available – utilization of POSIX interface on any FUSE-compatible Unix system [18] is possible with a FUSE-based file system provided by iRODS. The iRODS system provides built-in support for federalization through a Zone mechanism. A Zone is a single iRODS installation. Each iRODS Zone is a separate entity that can be managed in a different way and can include different storage resources. To access data located in another Zone, dedicated user accounts can be created in a remote Zone with a pointer to the home Zone of the user.

Although iRODS is a powerful and flexible tool that allows for connecting organizationally distributed installations, it also has some drawbacks. iRODS does not provide location transparency for data stored across multiple federated iRODS installations. Users manage the data location by themselves, so data-access transparency is not maintained.

Parrot [57] allows for attaching existing programs to remote data-management systems, which expose other access protocols (e.g., HTTP, FTP or XRootD) through the file system interface. Parrot utilizes a ptrace debugging interface to trap the system calls of a program and replace them with remote I/O operations. As a result, remote data can be accessed in the same way as local files. Unfortunately, the performance of Parrot is limited, as ptrace can generate significant overhead [57]. As a result, Parrot is not suited well for data-intensive applications.

Other interesting solutions are Syndicate Drive [55] and Storj [54]. Syndicate Drive is a virtual cloud storage system that combines the advantages of local storage, cloud storage, commodity CDNs, and network caches. Storj is a peer-to-peer cloud-storage network implementing end-to-end encryption to allow users to transfer and share data without support from a third-party data provider. Although these solutions contain algorithms that speed up data access, they are rarely used in the computing infrastructures of common use. The necessity of a data download before usage is a drawback, since such a preparation should be done in the background.

It is also worth mentioning FAX (Federating ATLAS storage systems using XrootD) [19]. FAX brings Tier 1, Tier 2, and Tier 3 storage resources together into a common namespace that is accessible anywhere. FAX Client software tools (e.g., ROOT, xrdcp) are able to reach the data regardless of their location. The N2N component that is responsible for mapping the global names to physical file names may be a performance bottleneck because of the LFC usage.

To enable the discovery and identification of data sets, open access services such as DataCite [10] or OpenAIRE [46] can be used. These services rely on standards such as OAI-PMH [59] for integration with the existing platforms for publication metadata harvesting and identify datasets through globally unique handles such as DOI [29] or PID. However, these services do not directly address the issue of accessing the underlying data by end users.

The tools described in this section are high-level data management-oriented. The standard POSIX file system interface is arguably a preferable interface for most applications. For this reason, the effort undertaken to abstract any specific interface with the POSIX interface is appreciated by the users. However, the main goal of these systems is to enable data access from anywhere in a uniform way rather than achieving high performance. Hence, despite the comfort of data access and management that they offer, their usage can hardly be limited in environments for HPC application execution.

4.5. Existing solution summary

Existing data-access solutions have several interesting features:

- data access anytime from anyplace with location transparency (Dropbox),
- increase of storage-system efficiency for chosen application on demand (QStorMan),
- increase of efficiency of data access due to use of dedicated hardware (PanFS),
- efficient work with many clients due to multiplication of components (BeeGFS),
- fast and reliable data transfer between providers due to use of efficient protocols and transfer supervision (Globus Connect),
- geographic replication that results in disaster protection and reduced risk of data loss (WOS),
- stable and efficient work with data when network is slow or unreliable due to client-side caching and strict support for disconnected mode (Coda),
- high flexibility in storage configuration due to built-in tiering (Scality Ring),
- dynamic adaptation and adjusting data-management behavior to administrator/user needs due to rules subsystem (iRODS),
- support for distributed data management in federations through Zones mechanism (iRODS),
- creating single view on independent data sources due to ability to attach remote data-management systems (Parrot),
- integration with various storage systems due to use of plug-ins (iRODS),
- discovery and identification of data sets (DataCite).

The features offered by the existing tools allow for fulfilling the requirements of different users. However, all of the solutions have their drawbacks (see Tables 1 and 2). None of the analyzed solutions has all of the following features:

- transparent data access anytime from anyplace,
- high efficiency and scalability,
- distributed data management.

To the best of the author's knowledge, none of the existing services or tools combines all three of the listed elements. The existing initiatives still have drawbacks. For example, NDS (National Data Storage) [44] – a national initiative aimed at developing and deploying a geographically distributed storage system with data backup and archiving service lacks ease of deploying and scalability, while DataNet Federation Consortium (DFC) [11] based on iRODS has drawbacks described in the previous chapter.

Table 1
Existing solution characteristics – part 1

Solution type	Data access solution	Characteristics	Drawbacks
Common Grid/Cloud data access tools	Lustre	High-performance cluster solution	Use of several tools together is needed – no data access transparency
	LFC	Provide common file names within grid or cloud	
	Swift		
	QStorMan	Improves data access performance on demand	
Tools for anytime/ anyplace data access	Dropbox	Easy to use	Limits on storage size and transfer speed
	Onedrive		
	Google Drive		
	ownCloud		
Tools for fast data movement	Globus Connect	Provides fast data movement and **data-sharing** capabilities based on GridFTP	Does not abstract access to data resources
High-performance parallel file systems	BeeGFS	High availability and performance due to scalable multithreaded core components with native Infiniband support	Designed to be used by single organization – no support for organizationally distributed environments
	GlusterFS	Scalability and performance due to elastic hashing algorithm	
	Coda	High availability due to strict support for disconnected mode operations	
	PanFS	High performance due to combination of functionality of parallel file system, volume manager and RAID engine into one holistic platform	
Solutions based on object storage	CephFS	POSIX-compliant distributed file system	
	DDN's WOS	True object storage	
	Scality Ring	Software-only storage solution with built-in tiering that provides high flexibility in storage configuration	

Table 2
Existing solution characteristics – part 2

Solution type	Data access solution	Characteristics	Drawbacks
Tools for map-reduce	HDFS	Designed to stream large data sets at high bandwidth to user's processes	No support for organizationally distributed environments
	Tachyon	Provides high performance for map-reduce applications using memory aggressively	
Tools for unified view of multi--organizational data	iRODS	Integrate various external data management systems using metadata catalogue – iCAT	Performance is not sufficiently high for data-intensive applications
	Parrot	Allows attaching existing programs to remote **data-management** systems through file system interface using ptrace debugging interface	
	Syndicate Drive	Base on data download before usage	
	Storj		
	FAX	Brings Tier 1, Tier 2 and Tier 3 storage resources together into a common namespace	
	DataCite	Enable discovery and identification of data sets	
	OpenAIRE		

5. Transparent data access with context awareness

The fact of the organizational distribution of the environment should not justify the efficiency problems, necessity of manual migrations, data staging, etc. Currently, very little data is being shared [5] because the sharing of data that is also being processed by data-intensive applications requires a lot of effort, including manual replication and migration of the data. Thus, the best features of the existing tools must be merged and extended to offer users a new quality of work. However, due to the large number of possible features of data-access systems required by different users (e.g., consistent view on distributed data, efficient data reading and/or writing, and avoiding or demanding data redundancy), it is impossible to choose a single set of mutually non-exclusive features that satisfy all stakeholders (compare to the CAP theorem [20]).

Typically, when a user plans to store data or gain access to data, he/she is faced with several aspects to be solved; for instance, a kind of data-access tool and/or storage system and/or localization to be used. Making a decision, the user has to take into account not only his/her needs but also the state of the environment (e.g., availability of storage space, its load and type, number of storage users, etc). It is inconvenient or even impossible for common users due to their lack of knowledge. The decision also requires efforts for limiting the negative impact of data distribution and the consequent provision of high availability [43], while HPC and/or HTC support [45] can require the use of data replication and analysis of information about delays to various data replicas during data access. While some existing tools cover selected parts of the above-mentioned aspects, it is still required that the user knows the different tools that must be used in different situations.

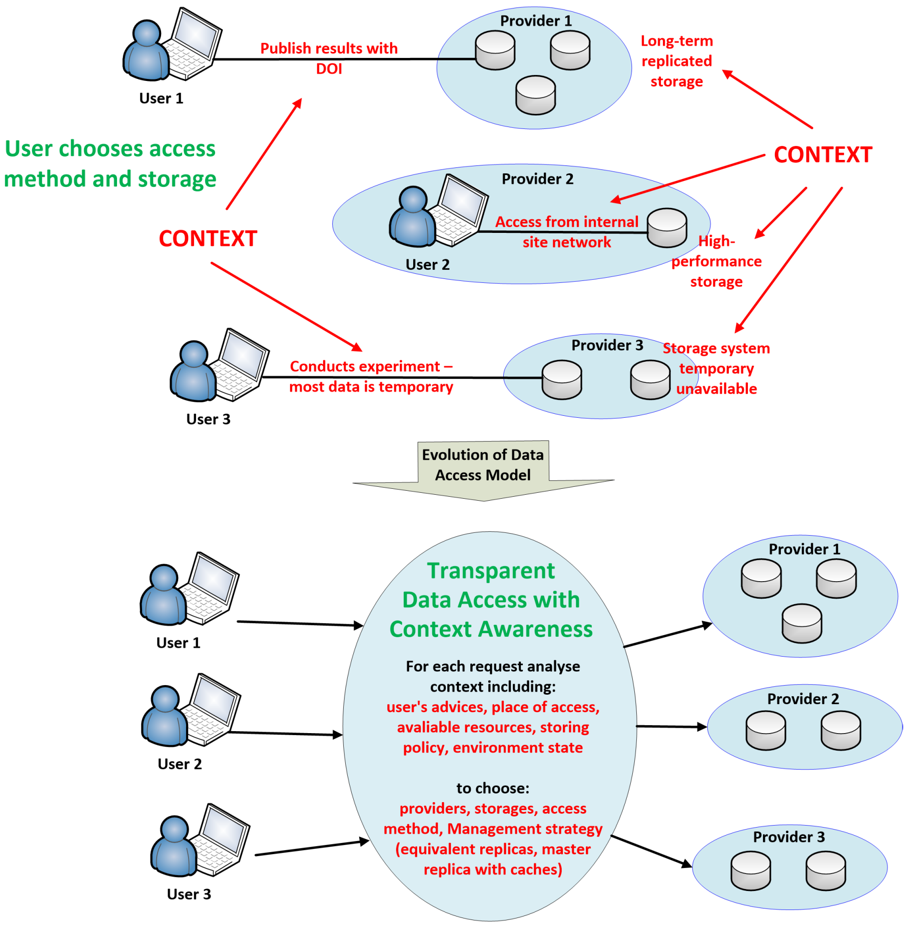

Figure 1. Proposed evolution of data-access model

After an analysis of the existing solutions and data-access stakeholder needs, the authors propose a model of Transparent Data Access with Context Awareness that automatically suites data-access characteristics based on data-access context (see Figure 1). The key feature of the model is determining the accessing and storing method automatically (suited to the user's needs) concerning the particular type of data-access requests (e.g., ACID or BASE [58]) or levels of reliability and efficiency. This automation is possible due to the context awareness – a property that allows the entity to both sense and react based on the environment state [1]. Thus, the second key element of the model is the ability to observe the environment and use the results of this observation to build its internal knowledge that is its basis for decision making. Since the selection can hardly be simplified to only one of the available options, the balance between options that provides the best effect is always sought. Such an approach is similar to the modern use of the CAP theorem that assumes a combination of the three features rather than arbitrarily selecting only two [7].

5.1. Roadmap

The data-access systems use metadata to describe various bits of information connected with data access; for example, user specific information (e.g., access control) and storage-specific information (e.g., location of data replicas) [14,64]. The metadata can also be used to describe the context of data access; e.g., the load of the data-access system components. The amount of metadata needed to describe the context usually grows along with environment complexity. The more context information is taken into account by the management algorithms, the more quality that can be provided to the user. However, the research indicates that operations on metadata are very likely to cause a bottleneck [13]. Thus, the first step is to organize the metadata to allow for the avoidance of bottlenecks. For this purpose, the metadata is categorized and grouped into classes with different consistency and synchronization models [65].

The appropriate metadata that describes the data-access context is the basis for a model that shows metadata processing and allows for the provision of the data-access method that is best-suited for a particular context. The model that is currently under the development consists of two elements: the levels that describe the roles of particular modules of context (described by the metadata) to provide specific functionality, and an algorithm that describes the linking of modules to provide flexible easy-to-use data access. The levels cover such aspects of data access as the unification and virtualization of access to data stored in different storage systems, performance of access operations, data-access simplicity, security and reliability of data storage, and the choice/provision of different types of quality and elastic management without central points.

Regardless of model development, the algorithms used by the different model levels are elaborated. These algorithms are tested using Onedata [62], a global data--access system that allows for access to data distributed at resources of independent organizations. It is expected that the final version of the Onedata software will implement the full model of Transparent Data Access with Context Awareness.

6. Conclusions

Applying the principles of open data research, [41] is an important factor accelerating the production and analysis of scientific results and worldwide collaboration. However, it requires easy-to-use data access and management solutions that changes their characteristics according to a particular context. Existing data-access tools have functionalities that are limited to particular use cases. To use the existing resources optimally, users are still required to gain a lot of knowledge about the data-access tools and infrastructures. It is time to evolve the data-access model towards a more convenient one. Thus, the authors are developing the model of Transparent Data Access with Context Awareness that automatically suites data-access characteristics to the user's needs based on the data-access context. As the model contains several levels that cover various problems, its detailed description will be published in a dedicated article.

Acknowledgements

The authors (RGS and JK) acknowledge the AGH-UST statutory grant. Support by the EOSC-hub project no. RI 777536 is also acknowledged. The authors owe special thanks to ACC Cyfronet AGH for providing the computing and storage infrastructure for this research.

References

[1] Abowd G.D., Dey A.K., Brown P.J., Davies N., Smith M., Steggles P.: Towards a Better Understanding of Context and Context-Awareness. In: *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC'99, pp. 304–307. Springer-Verlag, London, 1999. `http://dl.acm.org/citation.cfm?id=647985.743843`.

[2] Ananthakrishnan R., Chard K., Foster I., Tuecke S.: Globus Platform-as-a--Service for Collaborative Science Applications, *Concurrency and Computations: Practice and Experience*, vol. 27(2), pp. 290–305, 2014.

[3] BeeGFS. `http://http://www.beegfs.com/content/`.

[4] Baud J.P.B., Caey J., Lemaitre S., Nicholson C., Smith D., Stewart G.: LCG Data Management: From EDG to EGEE. In: *UK e-Science All Hands Meeting, Nottingham, UK*. 2005. `http://ppewww.ph.gla.ac.uk/preprints/2005/06/`.

[5] Borgman C.L.: The Conundrum of Sharing Research Data, *Journal of the American Society for Information Science and Technology*, vol. 63(6), pp. 1059–1078, 2012. `http://dx.doi.org/10.1002/asi.22634`.

[6] Braam P.J.: The Coda Distributed File System. `http://www.coda.cs.cmu.edu/ljpaper/lj.html`.

[7] Brewer E.: CAP twelve years later: How the "Rules" have changed, *Computer*, vol. 45(2), pp. 23–29, 2012. `http://dx.doi.org/10.1109/MC.2012.37`.

[8] Ceph Filesystem. `http://ceph.com/docs/next/cephfs/`.

[9] Chen S., Wang Y., Pedram M.: A joint optimization framework for request scheduling and energy storage management in a data center. In: Pu C., Mohindra A. (eds.), *8th IEEE International Conference on Cloud Computing, CLOUD 2015*, pp. 163–170, IEEE Computer Society, 2015. http://dblp.uni-trier.de/db/conf/IEEEcloud/IEEEcloud2015.html#ChenWP15.

[10] DataCite: DataCite : helping you to find, access, and reuse research data, 2011. http://datacite.org.

[11] DataNet Federation Consortium. http://datafed.org/.

[12] Dhar V.: Data science and prediction, *Communications of the ACM*, vol. 56(12), pp. 64–73, 2013. http://dx.doi.org/10.1145/2500499.

[13] Dong D., Herbert J.: FSaaS: File System as a Service. In: *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual.*

[14] Dutka Ł., Wrzeszcz M., Lichoń T., Słota R., Zemek K., Trzepla K., Opioła Ł., Słota R., Kitowski J.: Onedata – a step forward towards globalization of data access for computing infrastructures, *Procedia Computer Science*, vol. 51, pp. 2843–2847, 2015. http://dx.doi.org/10.1016/j.procs.2015.05.445.

[15] EGI-Engage. https://www.egi.eu/about/egi-engage/.

[16] European Commission, Public consultation 'Science 2.0': science in transition. http://ec.europa.eu/research/consultations/science-2.0/background.pdf.

[17] Foster I., Kesselman C.: *The Grid. Blueprint for a new computing infrastructure*, Morgan Kaufmann Publishers, San Francisco, USA, 1999.

[18] FUSE: Filesystem in Userspace. http://fuse.sourceforge.net/.

[19] Gardner R., Campana S., Duckeck G., Elmsheuser J., Hanushevsky A., Hönig F.G., Iven J., Legger F., Vukotic I., Yang W.: Data federation strategies for ATLAS using XRootD, *Journal of Physics: Conference Series*, vol. 513, p. 042049, 2014. http://dx.doi.org/10.1088/1742-6596/513/4/042049.

[20] Gilbert S., Lynch N.: Brewer's conjecture and the feasibility of consistent available partition-tolerant web services, *ACM SIGACT News*, vol. 33(2), pp. 51–59, 2002. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.1495&rep=rep1&type=pdf.

[21] GlusterFS community website. http://www.gluster.org/about/.

[22] Grid File Access Library 2.0 official page. https://svnweb.cern.ch/trac/lcgutil/wiki/gfal2.

[23] Han L., Huang H., Xie C.: Performance Analysis of NAND Flash Based Cache for Network Storage System. In: *IEEE Eighth International Conference on Networking, Architecture and Storage, NAS 2013*, pp. 68–75, IEEE Computer Society, 2013. http://dblp.uni-trier.de/db/conf/nas/nas2013.html#HanHX13.

[24] Hayashi C.: What is Data Science? Fundamental Concepts and a Heuristic Example. In: *Data Science, Classification, and Related Methods. Studies in Classification, Data Analysis, and Knowledge Organization*, 1998.

[25] Hey T., Tansley S., Tolle K. (eds.): *The Fourth Paradigm: Data-Intensive Scientific Discovery*, 2009. http://research.microsoft.com/en-us/collaboration/fourthparadigm/4th_paradigm_book_complete_lr.pdf.

[26] Human Brain Project. https://www.humanbrainproject.eu/.

[27] Hünich D., Müller-Pfefferkorn R.: Managing large datasets with iRODS – a performance analyses. In: *Proceedings of the International Multiconference on Computer Science and Information Technology, IMCSIT 2010*, pp. 647–654. 2010. http://dblp.uni-trier.de/db/conf/imcsit/imcsit2010.html#HunichM10.

[28] Indigo DataCloud. https://www.indigo-datacloud.eu/.

[29] International DOI Foundation: *DOI Handbook*, 2012. http://dx.doi.org/10.1000/182.

[30] Kryza B., Dutka L., Slota R., Kitowski J.: Dynamic VO establishment in distributed heterogeneous business environments. In: Allen G., Nabrzyski J., Seidel E., van Albada G.D., Dongarra J.J., Sloot P.M.A. (eds.): *ICCS (2)*, *Lecture Notes in Computer Science*, vol. 5545, pp. 709–718. Springer, 2009. http://dblp.uni-trier.de/db/conf/iccS/iccS2009-2.html#KryzaDSK09.

[31] Lamanna G., Antonelli L.A., Contreras J.L., Knödlseder J., Kosack K., Neyroud N., Aboudan A., Arrabito L., Barbier C., Bastieri D., Boisson C., Brau-Nogué S., Bregeon J., Bulgarelli A., Carosi A., Costa A., De Cesare G., de los Reyes R., Fioretti V., Gallozzi S., Jacquemier J., Khelifi B., Kocot J., Lombardi S., Lucarelli F., Lyard E., Maier G., Massimino P., Osborne J.P., Perri M., Rico J., Sanchez D.A., Satalecka K., Siejkowski H., Stolarczyk T., Szepieniec T., Testa V., Walter R., Ward J.E., Zoli A.: Cherenkov Telescope Array Data Management. In: *Proceedings of the 34th International Cosmic Ray Conference (ICRC2015), The Hague, The Netherlands, 2015*, pp. 1–8, 2015.

[32] Lustre. http://www.whamcloud.com/lustre/.

[33] Martin A.: OneDrive vs Google Drive vs Dropbox: The best cloud storage service of 2017? http://www.alphr.com/dropbox/7034/onedrive-vs-google-drive-vs-dropbox-the-best-cloud-storage-service-of-2017.

[34] Martini B., Choo R.K.K.: Cloud storage forensics: ownCloud as a case study, *Digital Investigation*, vol. 10(4), pp. 287–299, 2013. http://dblp.uni-trier.de/db/journals/di/di10.html#MartiniC13.

[35] Memon A.S., Jensen J., Cernivec A., Benedyczak K., Riedel M.: Federated authentication and credential translation in the EUDAT collaborative data infrastructure. In: *Proceedings of the 7th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2014, London, United Kingdom*, pp. 726–731, IEEE Computer Society, 2014. http://dblp.uni-trier.de/db/conf/ucc/ucc2014.html#MemonJCBR14.

[36] Mills S., Lucas S., Irakliotis L., Rappa M., Carlson T., Perlowitz B.: DEMYSTIFYING BIG DATA: A Practical Guide to Transforming the Business of Government. Technical report. https://bigdatawg.nist.gov/_uploadfiles/M0068_v1_3903747095.pdf (accessed 13.06.2018).

[37] Molloy J.C.: The Open Knowledge Foundation: open data means better science, *PLoS Biology*, vol. 9(12), p. e1001195, 2011. http://dx.doi.org/10.1371/journal.pbio.1001195.

[38] Nikolow D., Slota R., Polak S., Mitera D., Pogoda M., Winiarczyk P., Kitowski J.: Model of QoS management in a distributed data sharing and archiving system. In: Alexandrov V.N., Lees M., Krzhizhanovskaya V.V., Dongarra J., Sloot P.M.A. (eds.), *ICCS*, *Procedia Computer Science*, vol. 18, pp. 100–109, Elsevier, 2013. http://dblp.uni-trier.de/db/conf/iccS/iccS2013.html#NikolowSPMPWK13.

[39] OpenStack Object Storage ("Swift"). https://wiki.openstack.org/wiki/Swift.

[40] Pacheco L., Halalai R., Schiavoni V., Pedone F., Rivière E., Felber P.: GlobalFS: A strongly consistent multi-site file system. In: *2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS)*, pp. 147–156, 2016.

[41] Pampel H., Dallmeier-Tiessen S.: *Open research data: from vision to practice*, pp. 213–224. Springer International Publishing, Cham, 2014. http://dx.doi.org/10.1007/978-3-319-00026-8_14.

[42] PanFS Storage Operating System. http://www.panasas.com/products/panfs.

[43] Piedad F., Hawkins M.: *High availability: design, techniques, and processes*, Prentice Hall, 2001.

[44] Polish National Data Storage. https://www.elettra.eu/Conferences/2014/BDOD/uploads/Main/Polish%20National%20Data%20Storage.pdf.

[45] Raicu I.: *Many-task computing: bridging the gap between high throughput computing and high performance computing*, VDM Verlag, 2009.

[46] Rettberg N., Principe P.: Paving the way to Open Access scientific scholarly information: OpenAIRE and OpenAIREplus. In: Baptista A.A., Linde P., Lavesson N., de Brito M.A. (eds.), *International Conference on Electronic Publishing, ELPUB*, IOS Press, 2012. http://dblp.uni-trier.de/db/conf/elpub/elpub2012.html#RettbergP12.

[47] Röblitz T.: Towards implementing virtual data infrastructures – a case study with iRODS, *Computer Science*, vol. 13(4), pp. 21–34, 2012. http://dblp.uni-trier.de/db/journals/aghcs/aghcs13.html#Roblitz12.

[48] Scality. http://www.scality.com/products/what-is-ring/.

[49] Shafer J., Rixner S., Cox A.: The Hadoop distributed filesystem: Balancing portability and performance. In: *2010 IEEE International Symposium on Performance Analysis of Systems Software (ISPASS)*, pp. 122–133, 2010. http://dx.doi.org/10.1109/ISPASS.2010.5452045.

[50] Słota R.: Storage QoS provisioning for execution programming of data-intensive applications, *Scientific Programming*, vol. 20(1), pp. 69–80, 2012. http://dblp.uni-trier.de/db/journals/sp/sp20.html#Slota12.

[51] Słota R., Król D., Skałkowski K., Orzechowski M., Nikolow D., Kryza B., Wrzeszcz M., Kitowski J.: A toolkit for Storage QoS provisioning for data--intensive applications, *Computer Science*, vol. 13(1), pp. 63–73, 2012. http://dblp.uni-trier.de/db/journals/aghcs/aghcs13.html#SlotaOSONKWK12.

[52] Słota R., Nikolow D., Kitowski J., Król D., Kryza B.: FiVO/QStorMan semantic toolkit for supporting data-intensive applications in distributed environments, *Computing and Informatics*, vol. 31(5), pp. 1003–1024, 2012. http://dblp.uni-trier.de/db/journals/cai/cai31.html#SlotaNKOK12.

[53] Słota R., Nikolow D., Skałkowski K., Kitowski J.: Management of data access with quality of service in PL-Grid environment, *Computing and Informatics*, vol. 31(2), pp. 463–479, 2012. http://dblp.uni-trier.de/db/journals/cai/cai31.html#SlotaNSK12.

[54] Storj. http://storj.io/.

[55] Syndicate drive. http://syndicatedrive.com/.

[56] Tachyon Project. http://tachyon-project.org/.

[57] Thain D., Livny M.: Parrot: An application environment for data-intensive computing, *Scalable Computing: Practice and Experience*, vol. 6(3), pp. 9–18, 2005.

[58] Tudorica B.G., Bucur C.: A comparison between several NoSQL databases with comments and notes. In: *Roedunet International Conference (RoEduNet)*, 2011.

[59] Van de Sompel H., Nelson M., Lagoze C., Warner S.: Resource harvesting within the OAI-PMH framework, *D-Lib Magazine*, vol. 10(12), 2004. http://www.dlib.org/dlib/december04/vandesompel/12vandesompel.html.

[60] Web Object Scaler. http://www.ddn.com/products/object-storage-web-object-scaler-wos/#aboutwos.

[61] Weil S.A., Leung A.W., Brandt S.A., Maltzahn C.: RADOS: A scalable, reliable storage service for petabyte-scale storage clusters. https://ceph.com/wp-content/uploads/2016/08/weil-rados-pdsw07.pdf.

[62] What is Onedata? https://onedata.org/docs/doc/getting_started/what_is_onedata.html.

[63] Worldwide LHC Computing Grid. http://wlcg.web.cern.ch/.

[64] Wrzeszcz M., Trzepla K., Słota R., Zemek K., Lichoń T., Opiola Ł., Nikolow D., Dutka Ł., Słota R., Kitowski J.: Metadata organization and management for globalization of data access with onedata. In: *Parallel Processing and Applied Mathematics – 11th International Conference, PPAM 2015, Krakow, Poland, September 6–9, 2015. Revised Selected Papers, Part I*, pp. 312–321, 2015. http://dx.doi.org/10.1007/978-3-319-32149-3_30.

[65] Wrzeszcz M., Nikolow D., Lichoń T., Słota R., Dutka Ł., Słota R.G., Kitowski J.: Consistency models for global scalable data access services. In: Wyrzykowski R., Dongarra J., Deelman E., Karczewski K.: *12-th International Conference on Parallel Processing and Applied Mathematics, PPAM 2017, Lublin, Poland, September 2017*, Lecture Notes in Computer Science.

Affiliations

Michał Wrzeszcz
AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, al. Mickiewicza 30, 30-059 Krakow, Poland; Academic Computer Center CYFRONET AGH, Nawojki 11, 30-950 Krakow, Poland; wrzeszcz@agh.edu.pl

Renata G. Słota
AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, al. Mickiewicza 30, 30-059 Krakow, Poland, rena@agh.edu.pl

Jacek Kitowski
AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, al. Mickiewicza 30, 30-059 Krakow, Poland; Academic Computer Center CYFRONET AGH, Nawojki 11, 30-950 Krakow, Poland, kito@agh.edu.pl

Received: 28.03.2018
Revised: 25.04.2018
Accepted: 25.04.2018